Implementation exercises for the course Heuristic Optimization

Christian L. Camacho-Villalón
ccamacho@ulb.ac.be

IRIDIA, CoDE, ULB

February 29, 2024

Exercise 1.1: Iterative Improvement for the LOP

Goal: Implement perturbative local search algorithms for the LOP

1. Linear Ordering Problem (LOP)
2. First-improvement and Best-Improvement
3. Transpose, exchange and insert neighborhoods
4. Random initialization vs. CW heuristic
5. Statistical Empirical Analysis

The Linear Ordering Problem (1/3)

- *Ranking in sport tournaments*: establishing the order in which criteria such as wins, losses, draws, points scored, goals scored, or head-to-head results should be considered
- *Archeology*: establishing a chronological sequence of events or cultural phases based on the relative ordering of archaeological artifacts
- *Economics*: triangularization of input-output matrices allow to establish the interdependencies between different sectors of an economy
- etc.

Linear Ordering Problem (2/3)

Given

An n x n matrix C, where the value of row i and column j is noted c_{ij}.

	1	2	3	4
1	c_{11}	c_{12}	c_{13}	c_{14}
2	c_{21}	c_{22}	c_{23}	c_{24}
3	c_{31}	c_{32}	c_{33}	c_{34}
4	c_{41}	c_{42}	c_{43}	c_{44}

Objective

Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value $f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

Linear Ordering Problem (2/3)

Given

An n x n matrix C, where the value of row i and column j is noted c_{ij}.

	1	2	3	4
1	c_{11}	c_{12}	c_{13}	c_{14}
2	c_{21}	c_{22}	c_{23}	c_{24}
3	c_{31}	c_{32}	c_{33}	c_{34}
4	c_{41}	c_{42}	c_{43}	c_{44}

Objective

Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value $f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

Linear Ordering Problem example (3/3)

$$\pi = (1, 2, 3, 4)$$

$$C = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 3 & 2 & 4 \\ 2 & 2 & 1 & 1 & 3 \\ 3 & 1 & 2 & 2 & 1 \\ 4 & 4 & 5 & 1 & 4 \end{array}$$

$$f(\pi) = 3 + 2 + 4 + 1 + 3 + 1 = \mathbf{14}$$

Objective

Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value $f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

Linear Ordering Problem example (3/3)

$$\pi = (1, 2, 3, 4) \qquad C = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 3 & 2 & 4 \\ 2 & 2 & 1 & 1 & 3 \\ 3 & 1 & 2 & 2 & 1 \\ 4 & 4 & 5 & 1 & 4 \end{array}$$

$$f(\pi) = 3 + 2 + 4 + 1 + 3 + 1 = \mathbf{14}$$

Objective

Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value $f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

Linear Ordering Problem example (3/3)

$$\pi = (1, 2, 3, 4) \qquad C = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 3 & 2 & 4 \\ 2 & 2 & 1 & 1 & 3 \\ 3 & 1 & 2 & 2 & 1 \\ 4 & 4 & 5 & 1 & 4 \end{array}$$

$$f(\pi) = 3 + 2 + 4 + 1 + 3 + 1 = \mathbf{14}$$

Objective

Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value $f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

Linear Ordering Problem example (3/3)

$$\pi = (1, 2, 3, 4) \qquad C = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 3 & 2 & 4 \\ 2 & 2 & 1 & 1 & 3 \\ 3 & 1 & 2 & 2 & 1 \\ 4 & 4 & 5 & 1 & 4 \end{array}$$

$$f(\pi) = 3 + 2 + 4 + 1 + 3 + 1 = \mathbf{14}$$

Objective

Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value $f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}$ is maximized.

Exercise 1.1: Iterative Improvement for the LOP

Implement 12 iterative improvements algorithms for the LOP

Exercise 1.1: Iterative Improvement for the LOP

Implement 12 iterative improvements algorithms for the LOP

- Pivoting rule:
  1. first-improvement
  2. best-improvement
- Neighborhood:
  1. Transpose
  2. Exchange
  3. Insert
- Initial solution:
  1. Random permutation
  2. Chenery and Watanabe heuristic

Exercise 1.1: Iterative Improvement for the LOP

Implement 12 iterative improvements algorithms for the LOP

- Pivoting rule:
  1. first-improvement
  2. best-improvement
- Neighborhood:
  1. Transpose
  2. Exchange
  3. Insert
- Initial solution:
  1. Random permutation
  2. Chenery and Watanabe heuristic

2 pivoting rules $\times$ 3 neighborhoods $\times$ 2 initialization methods $=$
12 combinations

Exercise 1.1: Iterative Improvement for the LOP

Implement 12 iterative improvements algorithms for the LOP

Do not implement 12 programs!

Reuse code and use command-line parameters

```
./lop11 --first --transpose --cw
./lop11 --best --exchange --random
etc.
```

Use a parameter to choose the instance file, for instance:

```
./lop11 -i <instance_file>
```

Exercise 1.1: Iterative Improvement for the LOP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Exercise 1.1: Iterative Improvement for the LOP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighbour to choose? Pivoting rule

- **Best Improvement:** choose best from all neighbours of π
  - ✔ Better quality
  - ✘ Requires evaluation of all neighbours in each step
- **First improvement:** evaluate neighbours in fixed order and choose first improving neighbour.
  - ✔ More efficient
  - ✘ Order of evaluation may impact quality / performance

Exercise 1.1: Iterative Improvement for the LOP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighbour to choose? Pivoting rule

- **Best Improvement:** choose best from all neighbours of π
  - ✔ Better quality
  - ✘ Requires evaluation of all neighbours in each step
- **First improvement:** evaluate neighbours in fixed order and choose first improving neighbour.
  - ✔ More efficient
  - ✘ Order of evaluation may impact quality / performance

Exercise 1.1: Iterative Improvement for the LOP

Iterative Improvement

$\pi := \texttt{GenerateInitialSolution()}$
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighbour to choose? Pivoting rule

- **Best Improvement:** choose best from all neighbours of π
  - ✔ Better quality
  - ✘ Requires evaluation of all neighbours in each step
- **First improvement:** evaluate neighbours in fixed order and choose first improving neighbour.
  - ✔ More efficient
  - ✘ Order of evaluation may impact quality / performance

Exercise 1.1: Iterative Improvement for the LOP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Initial solution

- Random permutation
- Chenery and Watanabe (CW) heuristic

Exercise 1.1: Iterative Improvement for the LOP

Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Chenery and Watanabe (CW) heuristic

Construct the solution by inserting **one row at a time**, always selecting the most "attractive" row: the one that maximizes the sum of elements having an influence on objective function.

The "attractiveness" of a row i at step s is: $\sum_{j=s+1}^{n} c_{\pi(i)j}$

Exercise 1.1: Iterative Improvement for the LOP

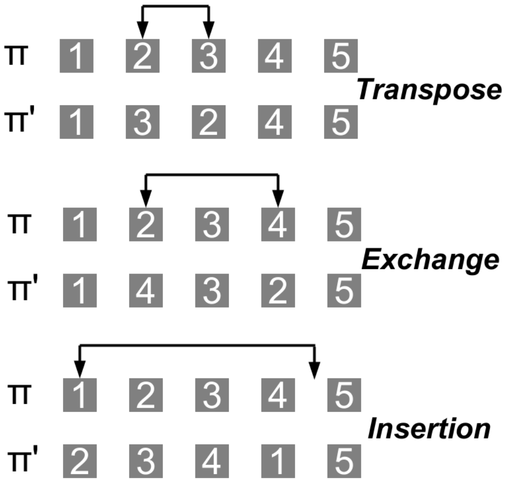
Iterative Improvement

$\pi :=$ `GenerateInitialSolution()`
while π is not a local optimum **do**
  choose a neighbour $\pi' \in \mathcal{N}(\pi)$ such that $F(\pi') < F(\pi)$
  $\pi := \pi'$

Which neighborhood $\mathcal{N}(\pi)$?

- Transpose
- Exchange
- Insertion

Exercise 1.1: Iterative Improvement for the LOP



Exercise 1.1: Iterative Improvement for the LOP

π 1 2 3 4

π' 1 4 3 2

Example: Exchange π_i and π_j ($i \neq j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

$\pi =$

	1	2	3	4
1	1	**2**	**3**	**4**
2	5	6	**7**	**8**
3	9	10	11	**12**
4	13	14	15	16

$\pi' =$

	1	2	3	4
1	1	***4***	**3**	***2***
4	*13*	*16*	***15***	***14***
3	9	*12*	11	***10***
2	*5*	*8*	*7*	***6***

Only a subset of the changes affect the computation of the objective function
Do not recompute the evaluation function from scratch!
Equivalent speed-ups with Transpose and Insertion.

*(**NOTE**: Implementing speed-ups will get you extra points in the exercise)*

Exercise 1.1: Iterative Improvement for the LOP

π	1	2	3	4
π'	1	4	3	2

Example: Exchange π_i and π_j ($i \neq j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

$\pi =$

	1	2	3	4
1	1	**2**	**3**	**4**
2	5	6	**7**	**8**
3	9	10	11	**12**
4	13	14	15	16

$\pi' =$

	1	2	3	4
1	1	***4***	**3**	***2***
4	*13*	*16*	***15***	***14***
3	9	*12*	11	***10***
2	*5*	*8*	*7*	***6***

Only a subset of the changes affect the computation of the objective function
Do not recompute the evaluation function from scratch!
Equivalent speed-ups with Transpose and Insertion.

*(**NOTE**: Implementing speed-ups will get you extra points in the exercise)*

Exercise 1.1: Iterative Improvement for the LOP

π 1 2 3 4

π' 1 4 3 2

Example: Exchange π_i and π_j ($i \neq j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

$\pi =$

	1	2	3	4
1	1	**2**	**3**	**4**
2	5	6	**7**	**8**
3	9	10	11	**12**
4	13	14	15	16

$\pi' =$

	1	2	3	4
1	1	***4***	**3**	***2***
4	*13*	*16*	***15***	***14***
3	9	*12*	11	***10***
2	*5*	*8*	*7*	***6***

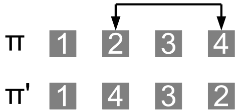
Only a subset of the changes affect the computation of the objective function

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion.

*(**NOTE**: Implementing speed-ups will get you extra points in the exercise)*

Exercise 1.1: Iterative Improvement for the LOP



Example: Exchange π_i and π_j ($i \neq j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

$\pi =$

	1	2	3	4
1	1	**2**	**3**	**4**
2	5	6	**7**	**8**
3	9	10	11	**12**
4	13	14	15	16

$\pi' =$

	1	2	3	4
1	1	***4***	**3**	***2***
4	*13*	*16*	***15***	***14***
3	9	*12*	11	***10***
2	*5*	*8*	*7*	***6***

Only a subset of the changes affect the computation of the objective function

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion.

*(**NOTE**: Implementing speed-ups will get you extra points in the exercise)*

Exercise 1.1: Iterative Improvement for the LOP

π 1 2 3 4

π' 1 4 3 2

Example: Exchange π_i and π_j ($i \neq j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

$\pi =$

	1	2	3	4
1	1	**2**	**3**	**4**
2	5	6	**7**	**8**
3	9	10	11	**12**
4	13	14	15	16

$\pi' =$

	1	2	3	4
1	1	***4***	**3**	***2***
4	*13*	*16*	***15***	***14***
3	9	*12*	11	***10***
2	*5*	*8*	*7*	***6***

Only a subset of the changes affect the computation of the objective function

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion.

*(**NOTE**: Implementing speed-ups will get you extra points in the exercise)*

Exercise 1.1: Iterative Improvement for the LOP

π	1	2	3	4
π'	1	4	3	2

Example: Exchange π_i and π_j ($i \neq j$), $\pi' = \texttt{Exchange}(\pi, i, j)$

$\pi =$

	1	2	3	4
1	1	**2**	**3**	**4**
2	5	6	**7**	**8**
3	9	10	11	**12**
4	13	14	15	16

$\pi' =$

	1	2	3	4
1	1	***4***	**3**	***2***
4	*13*	*16*	***15***	***14***
3	9	*12*	11	***10***
2	*5*	*8*	*7*	***6***

Only a subset of the changes affect the computation of the objective function

Do not recompute the evaluation function from scratch!

Equivalent speed-ups with Transpose and Insertion.

*(**NOTE**: Implementing speed-ups will get you extra points in the exercise)*

Exercise 1.1: Iterative Improvement for the LOP

Instances

- LOP instances with sizes 150 and 250.
- A full description is provided in the project document on TEAMS

Experiments

Apply each algorithm k once to each instance i and record its :

1. Relative percentage deviation $\Delta_{ki} = 100 \cdot \dfrac{\text{best-known}_i - \text{cost}_{ki}}{\text{best-known}_i}$
2. Computation time (t_{ki})

Note: use constant random seed across algorithms, for each instance

Report for each algorithm k

- Average relative percentage deviation
- Sum of computation time

Exercise 1.1: Iterative Improvement for the LOP

Instances

- LOP instances with sizes 150 and 250.
- A full description is provided in the project document on TEAMS

Experiments

Apply each algorithm k once to each instance i and record its :

1. Relative percentage deviation $\Delta_{ki} = 100 \cdot \frac{\text{best-known}_i - \text{cost}_{ki}}{\text{best-known}_i}$
2. Computation time (t_{ki})

Note: use constant random seed across algorithms, for each instance

Report for each algorithm k

- Average relative percentage deviation
- Sum of computation time

Exercise 1.1: Iterative Improvement for the LOP

Instances

- LOP instances with sizes 150 and 250.
- A full description is provided in the project document on TEAMS

Experiments

Apply each algorithm *k* once to each instance *i* and record its :

1. Relative percentage deviation $\Delta_{ki} = 100 \cdot \frac{\text{best-known}_i - \text{cost}_{ki}}{\text{best-known}_i}$
2. Computation time (t_{ki})

Note: use constant random seed across algorithms, for each instance

Report for each algorithm *k*

- Average relative percentage deviation
- Sum of computation time

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Statistical test

- Paired t-test
- Wilcoxon signed-rank test

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test. *Example:* For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.
  Typical values of α are 0.05 or 0.01.

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.
  Typical values of α are 0.05 or 0.01.

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.
  Typical values of α are 0.05 or 0.01.

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.
  Typical values of α are 0.05 or 0.01.

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (1)

- *Statistical hypothesis tests* are used to assess the validity of statements about properties of or relations between sets of statistical data.
- The statement to be tested (or its negation) is called the *null hypothesis* (H_0) of the test.
  Example: For the Wilcoxon signed-rank test, the null hypothesis is that 'the median of the differences is zero'.
- The *significance level* (α) determines the maximum allowable probability of incorrectly rejecting the null hypothesis.
  Typical values of α are 0.05 or 0.01.

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (2)

- The application of a test to a given data set results in a *p-value*, which represents the probability that the null hypothesis is incorrectly rejected.
- The null hypothesis is rejected iff this p-value is smaller than the previously chosen significance level.
- Most common statistical hypothesis tests are already implemented in statistical software such as the *R software environment* (http://www.r-project.org/).

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (2)

- The application of a test to a given data set results in a *p-value*, which represents the probability that the null hypothesis is incorrectly rejected.
- The null hypothesis is rejected iff this p-value is smaller than the previously chosen significance level.
- Most common statistical hypothesis tests are already implemented in statistical software such as the *R software environment* (http://www.r-project.org/).

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Background: Statistical hypothesis tests (2)

- The application of a test to a given data set results in a *p-value*, which represents the probability that the null hypothesis is incorrectly rejected.
- The null hypothesis is rejected iff this p-value is smaller than the previously chosen significance level.
- Most common statistical hypothesis tests are already implemented in statistical software such as the *R software environment* (http://www.r-project.org/).

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("lop-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("lop-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112 // Greater than 0.05!
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("lop-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("lop-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112 // Greater than 0.05!
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("lop-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("lop-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112 // Greater than 0.05!
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("lop-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("lop-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112 // Greater than 0.05!
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("lop-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("lop-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112 // Greater than 0.05!
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Exercise 1.1: Iterative Improvement for the LOP

Is there a statistically significant difference between the solution quality generated by the different algorithms?

Example in R

```
best.known <- read.table ("best-known.dat")
a.cost <- read.table("lop-best-ex-rand.dat")$V1
a.cost <- 100 * (a.cost - best.known) / best.known
b.cost <- read.table("lop-best-ins-rand.dat")$V1
b.cost <- 100 * (b.cost - best.known) / best.known
t.test (a.cost, b.cost, paired=T)$p.value
[1] 0.8819112 // Greater than 0.05!
wilcox.test (a.cost, b.cost, paired=T)$p.value
[1] 0.0019212
```

Exercise 1.2 VND algorithms for the LOP

Implement 2 VND algorithms for the LOP

- Pivoting rule: first-improvement
- Neighborhood order:
  1. transpose $\rightarrow$ exchange $\rightarrow$ insert
  2. transpose $\rightarrow$ insert $\rightarrow$ exchange
- Initial solution:
  1. CW heuristic

Exercise 1.2 VND algorithms for the LOP

Variable Neighbourhood Descent (VND)

k neighborhoods $\mathcal{N}_1, \ldots, \mathcal{N}_k$

$\pi := \texttt{GenerateInitialSolution()}$

$i := 1$

repeat

  choose the first improving neighbor $\pi' \in \mathcal{N}_i(\pi)$

  if $\nexists \pi'$ **then**

    $i := i + 1$

  else

    $\pi := \pi'$

    $i := 1$

until $i > k$

Exercise 1.2 VND algorithms for the LOP

Implement 4 VND algorithms for the LOP

- Instances: Same as for exercise 1.1
- Experiments: Same as for exercise 1.1 (i.e., **one run** of each algorithm per instance with constant seeds)
- Report: Same as for exercise 1.1
- Statistical tests: Same as for exercise 1.1

- Instances and “skeleton” code are available on TEAMS
- Some of the deliverables you need to provide in a zip folder with your name via TEAMS are:
  - your implementation in `C`, `C++` or `Java`. `Python` is **not** recommended
  - a README file explaining how to run your implementation from the command line on Linux
  - a report no longer than 5 pages describing the implementation of the algorithms and the results you obtained (more detail on TEAMS)
  - see the full description of the deliverables in the pdf on TEAMS
- Deadline is April 7, 2024 (23:59)
- Questions?
  Use the Post tab of the implementation channel on TEAMS